CONCURRENCY AND COMPUTATION: PRACTICE AND EXPERIENCE
Concurrency Computat.: Pract. Exper. 2015; **27**:3126–3144
Published online 5 February 2015 in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/cpe.3456

SPECIAL ISSUE PAPER

Role mining based on cardinality constraints

Ruixuan Li[1], Huaqing Li[1], Xiwu Gu[1,*,†], Yuhua Li[1,2], Wei Ye[1] and Xiaopu Ma[1]

[1]*School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, Hubei 430074, China*
[2]*State Key Laboratory of Software Engineering, Wuhan University, Wuhan 430072, China*

SUMMARY

Role mining was recently proposed to automatically find roles among user-permission assignments using data mining technologies. However, the current studies about role mining mainly focus on how to find roles, without considering the constraints that are essentially required in role-based access control systems. In this paper, we present a role mining algorithm with constraints, especially for the cardinality constraints. We illustrate it is essential for role mining to take cardinality constraints into account, and introduce the concepts of the cardinality constraints of roles and permissions. We further propose a role mining algorithm to generate roles based on these two kinds of cardinality constraints. The algorithm uses graph theory to model the role mining problem and maps the relation of two roles to the relation of graph elements. We set an optimization goal for role mining and employ graph optimization theory to find roles that satisfy the aforementioned cardinality constraints. We carry out the experiments to evaluate our approach. The experimental results demonstrate the rationality and effectiveness of the proposed algorithm. Copyright © 2015 John Wiley & Sons, Ltd.

Received 1 February 2013; Revised 14 June 2014; Accepted 24 November 2014

KEY WORDS: role-based access control (RBAC); role mining; cardinality constraints

1. INTRODUCTION

In recent years, the advantage of employing role-based access control (RBAC) model [1, 2] in application systems is so significant that more and more institutions want to transform their non-RBAC system to RBAC architecture. However, the cost of the transformation is exorbitant, especially for the process of role identification that finds out a role set that can represent the relationship of the user-permission assignments (UPAs). There are two ways for role identification: top–down approach and bottom–up approach. The top–down approach is based on business process and user scenario. It needs a large number of experts and spends much time on analyzing the business processes and user requirements to find out the functional roles and the related permission sets [3–7]. The process includes defining particular job functions, decomposing the functions into smaller units, and creating roles for these units by associating needed permissions. While the role sets established by the top–down approach have good characterizations of the real organizational structures, it is costly and time-consuming, and many organizations cannot afford it because the business processes may be complicated and the number of users and permissions in an organization may be very large.

*Correspondence to: Xiwu Gu, School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, Hubei 430074, China.
†E-mail: guxiwu@hust.edu.cn

The bottom–up approach is an automatic or semi-automatic way that identifies the role sets through analyzing the user-permission assignments using the technology of data mining, which is called *role mining*. Presently, there are already many role mining algorithms [8–11] that can automatically form role state with hierarchy or no hierarchy. In the early phase, the researchers enumerate all possible roles with the operation about the set [8, 9], such as cross and union. Later, researchers find out the mined roles do not accord with the actual hierarchy relation. Therefore, they propose some role mining algorithms with hierarchy [9, 10, 12]. However, few of the algorithms take the constraints into consideration. As we know, the constraints are the basic requirements for an RBAC model. The security of the RBAC systems will be enhanced through employing the constraints. For example, the mutual exclusion constraint can prevent the conflict permissions being put into one role. Generally, the mutual exclusion constraint will be naturally satisfied in the role mining process if the user-permission assignments are properly created. That is, if two permissions are mutual exclusion, they will not be assigned to the same user that makes them not be selected into one role using the current role mining algorithms. In this paper, we pay close attention to cardinality constraints because they are very common in the RBAC systems.

The cardinality constraint is one of constraints required by the RBAC model [2]. It has four situations. First, the number of user members owned by one role is limited. Second, the number of roles to which an individual user can belong should be limited. Third, the number of roles to which a permission can be assigned should also be restricted. Last, the number of permissions that a role can own should also have cardinality constraint. This paper focuses on the second and the third situations. The first situation can be simply solved by creating some roles with the same name or creating some roles with different names but the same functions. The fourth situation can also be simply solved by splitting the incompliant role into two or more roles. However, the second and the third situations cannot be solved simply.

The second situation is to avoid the number of roles that a user owns being too large. If the number is too large, the user will not be aware that he has some roles and the permissions owned by those roles after some time. That will lead the user to think he cannot do something but in fact he is able to. That is to say, it is difficult for the user to sift the roles if he has too many roles, which will increase personal management cost. In addition, if the number is too large, it will make the RBAC system degenerate to the direct permission assignment and the advantage of RBAC model will be limited. The third situation is to control the distribution of powerful permissions. The powerful permission should not be assigned to too many roles.

In this paper, we propose a role mining algorithm based on the second and third constraints. Our algorithm improves the graph optimization theory proposed by Zhang et al. in [11] to create role sets, considering the aforementioned constraints at the same time. We analyze the drawbacks of the original graph optimization theory and discuss our algorithm in four scenarios. The first scenario is two roles have the same permissions. The second scenario is the permission sets of the first role are a subset or superset of the permission sets of the second role. The third scenario is the permissions of two roles are crossed, and the last scenario is the permissions of two roles do not have any relation. In each scenario, we do different operations to get the results that satisfy the aforementioned two cardinality constraints. To establish the role hierarchy satisfying these constraints, we employ the graph theory and map the role element to graph element. Graph elements not only have the node information but also have the information about edges that form the hierarchy of the nodes. The criteria for a good role state are the minimal assignment cost and interpretability [13]. We employ the weighted structural complexity (WSC) [12] as the optimization function to measure the assignment cost. We use graph optimization theory to acquire the role hierarchy in the aforementioned four scenarios. We carry out the experiments on the real user-permission assignment relations and the simulated constraints on these relations. The experimental results demonstrate the effectiveness of the proposed algorithm.

Our contributions include three aspects. First, we introduce cardinality constraints into role mining algorithms, which enable RBAC systems to satisfy the real application requirements better. Second, we correct the drawback of the approach proposed in [11], in which it finds some roles but these roles cannot be assigned to any user. Last, we not only improve the accuracy of the result compared with graph optimization (GO) algorithm but also reduce the

DOI: 10.1002/cpe

number of roles compared with role-priority-based algorithm (RPA) and coverage of permissions-based algorithm (CPA) [14].

The remainder of this paper is organized as follows. We discuss related work in the role mining field in Section 2 and introduce the constraints of RBAC model including cardinality constraints in Section 3. We present our role mining algorithm with the second and third cardinality constraints in Section 4 and show the experimental results in Section 5. At last, we conclude the paper and discuss the future work in Section 6.

2. RELATED WORK

In 1995, role engineering was proposed by Coyne for building the RBAC system in the system without RBAC [3]. He introduced the top–down approach that starts from the business process and user scenario in order to find out a role state that reflects the authorization situation effectively, completely, and correctly. The approach requires a large number of domain experts to analyze the business processes and user scenarios. Because the number of business processes and users is too large, the way is time-consuming and human-intensive, which finally leads to the high cost that many small and medium companies cannot afford [7,15].

In order to overcome the aforementioned drawbacks, researchers introduced the bottom–up approach that automatically or semi-automatically finds out the role sets with data mining technology and graph theory. The approach called role mining starts from the user-permission assignments and some corresponding attribute information. Because of the low cost and simplicity, many researchers showed significant interests on this bottom–up approach. There are many role mining algorithms proposed in recent years. In 2005, Schlegelmilch and Steffens [10] first proposed ORCA, the OFFIS Role mining tool with Cluster Analysis, and its algorithm. It does a hierarchical clustering on permissions. The algorithm selects a maximal overlap pair each time and then uses the pair to update the remaining pairs. Each pair forms a role. The drawback of the ORCA is that each permission is discovered through only one path. In 2007, Zhang et al. proposed a role mining algorithm that is guided by the graph optimization theory [11,16]. It starts from the scenario that the permissions of each user form a role. They introduce an objective function for minimizing the number of edges and then divide the situations into equality, subset, and intersection. At last, they check whether each situation can improve the objective function or not, and respond according to the checking result. Similarly, Vaidya et al. propose a subset enumeration way to get the role sets. They introduced two algorithms: CompleteMiner and FastMiner [8].

In 2008, there are two role mining algorithms emerging. Molloy et al. [12] introduce the semantic information in the process of role mining to let the mined roles be better consistent with the organization's business process. The algorithm uses the formal concept analysis to find out the roles and it also offers an objective function called the weighted structural complexity (WSC) to guide the algorithm. The algorithm first generates the concept lattice and reduced lattice to get the role hierarchy and then clusters on the reduced lattice. Vaidya et al. [17] introduce the minimal perturbation into the role mining process to make the result be more close to the role sets that are already deployed. The algorithm first gets the candidate roles by the *FastMiner* algorithm. Then, it sorts these roles by the area that the role occupies in descending order. At last, according to the roles' order, it makes these roles cover the user-permission assignment until the user-permission table is entirely covered. Through this way, it can get a role sequence and then recommend the sequence for the system.

During the development of role mining, the researchers also give some other role mining algorithms. For example, [9] proposes a role mining algorithm using the frequent pattern mining. In [18], the authors map the basic role mining problem and its variants to the minimum tiling problem and its variants to mine the roles using the tiling databases theory. In their approach, the role mining problem is mapped to the discrete basis problem and then uses the theory to find out the roles. Vaidya et al. [19] transform the role mining problem to the Boolean matrix decomposition and then use the matrix decomposition theory to discover the roles. However, after 2008, the research about the algorithm study stops. Instead, people introduce some additional conditions into the role mining

DOI: 10.1002/cpe

process. For example, in [20], Ma et al. introduce the weight of the permission into role mining process; in [21], Takabi and Joshi introduce the similarity between the deployed roles and discovered roles; in [22], Molloy et al. think the UPA matrix may have noises. They propose to remove the noise before mining the roles.

Although the constraints are essential for the RBAC model, only a few researches took the constraints into account in role mining. In [23], Lu et al. consider the separation of duty (SoD) and exceptions. As mentioned earlier, the mutual exclusion constraint will be naturally satisfied in the role mining process if the user-permission assignments are properly created. Lu et al. allow redundant permissions in roles or redundant role assignments for users, which may lead to separation of duty (SoD) and exceptions. In such conditions, they use extended Boolean matrix decomposition to solve the problem. In [24,14], cardinality constraints are considered in role mining. However, [24] considers the maximum number of users that can be assigned to each role, that is the first situations of cardinality constraint. Although [14] also takes the second situations of cardinality constraint, it only considers single constraint. The combined effect on the mining process when many constraints are considered simultaneously needs be studied.

From the above analysis, we can see that the current research about constraints in role mining is very weak, especially in terms of multiple cardinality constraints. In this paper, our algorithm takes two cardinality constraints into consideration at the same time in role mining. Through the experiments, we know that our algorithm will not reduce the accuracy and not increase the WSC greatly while it takes the cardinality constraints into account. Our algorithm also produces the hierarchical output simultaneously. Hence, it enables the RBAC to reach the RBAC3's requirements, which are required by many application systems.

3. THE CONSTRAINT MODEL OF RBAC

In [2], a family of RBAC96 models was proposed. The family contains four members. Their relationship is shown in Figure 1. The RBAC0 is the basic model in the family, which defines from the two-level relationship of user-permission assignment to the three-level relationship of user-role-permission assignment. With the RBAC0, the security administrator can build the RBAC system from a system without RBAC. The RBAC1 increases the hierarchy on the basis of the RBAC0. The hierarchy relationship can reflect the position level in the organization. The RBAC2 increases the constraints on the basis of the RBAC0. These constraints make the system more secure. The RBAC3 combines the RBAC1 and the RBAC2. It offers role hierarchy and constraints.

In the family, the RBAC2 is an important complementary for the RBAC0 because constraint is an important function of RBAC, and some researchers think it is the main motivation of the RBAC appearance. Constraint is an effective mechanism for making the high-level organizational policy. It gives the security administrator convenience for role management. Especially, when the management of RBAC is not centralized, the high-level administrator can use it as a mandatory requirement to be imposed on the low-level administrators. Constraint can not only apply on the permission-role assignment (PA) and user-role assignment (UA) but also can apply on the user and role functions in the sessions. The common constraints are classified into six categories: mutually exclusive, cardinality constraints, prerequisite roles constraints, run-time mutually exclusive, number constraints of sessions, and time frequency constraints.

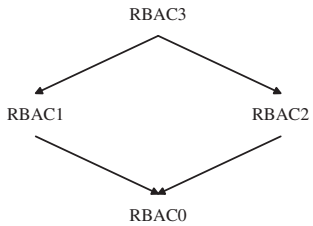


Figure 1. Relationship among role-based access control (RBAC) models.

DOI: 10.1002/cpe

Among these constraints, the latter three ones cannot be processed in the role mining process because they need run-time information that cannot be obtained in the role mining stage. The mutually exclusive can be automatically satisfied (illustrated in Section 1). This paper focuses on the cardinality constraints. We will consider the prerequisite roles constraints in the future work.

4. ROLE MINING BASED ON CARDINALITY CONSTRAINTS

4.1. Preliminaries

In this section, we define some concepts that will be used in the following sections. We adopt the National Institute of Standards and Technology standard of the RBAC model [25].

Definition 1. (RBAC)

- U, ROLES, OPS, and OBJ are the set of users, roles, operations, and objects.
- $UA \subseteq U \times ROLES$, a many-to-many mapping user-to-role assignment relation.
- PRMS (the set of permissions) $\subseteq \{(op, obj)\ op \in OPS \wedge obj \in OBJ\}$
- $PA \subseteq ROLES \times PRMS$, a many-to-many mapping of role-to-permission assignments.[1]
- $UPA \subseteq U \times PRMS$, a many-to-many mapping of user-to-permission assignments.
- assigned_users(R) = $\{u \in U\ (u, R) \in UA\}$, the mapping of role R onto a set of users.
- assigned_permissions(R) = $\{p \in PRMS\ (p, R) \in PA\}$, the mapping of role R onto a set of permissions.
- assigned_permissions(U) = $\{p \in PRMS\ (u, p) \in UPA\}$, the mapping of user U onto a set of permissions.
- assigned_users(P) = $\{u \in U\ (u, p) \in UPA\}$, the mapping of permission P onto a set of users.

Definition 2. (Cardinality constraints of roles)
The cardinality constraints of roles are defined as the maximum number of roles to which an individual user can belong. We will use the notation max_roles(u) to specify this constraint [26].

Definition 3. (Cardinality constraints of permissions)
The cardinality constraints of permissions are defined as the maximum number of roles to which a permission can belong. We will use the notation max_ roles (p) to specify this constraint.

Definition 4. (Rounded up)
The rounded up is defined as the smallest integer that is bigger than a specific double precision number. We will use the notation $\lceil\ \rceil$ to specify this rounded up, for example, $\lceil 0.5 \rceil = 1$ and $\lceil 1.0 \rceil = 1$.

4.2. The optimization metric

In [11], Zhang et al. propose a role mining algorithm using graph optimization and give an objective function that guides the optimization process. However, the algorithm has two drawbacks. Firstly, in the scenario that there is an overlap in the permission sets of two roles where neither role's permission sets are a subset of the other role's permission sets (Figure 2), a role containing the overlap permissions of the two roles is created and two links to this new role are created. Nevertheless, that is not enough, because the new role would not be assigned to any user. For instance, from Figure 2, we know that Steve and Bob have the permission sets {*p*1, *p*2, *p*3, *p*4, *p*5}, and now we get *r*1.*r*2 role, which has the permissions {*p*2, *p*3, *p*4, *p*5}. Because we do not have the role that contains the permission {*p*1}, the role *r*1.*r*2 cannot be assigned to any user. Secondly, the objective function only considers edge minimum and ignores many other factors, such as direct assignment. In view of that, we use the objective function proposed by [9] that is defined as follows.

Definition 5. (Weighted structural complexity)
Given $W = (wr, wu, wp, wh, wd)$, where $wr, wu, wp, wh, wd \in Q+ \cup \{\infty\}$, the WSC of an RBAC state γ, which is denoted as $wsc(\gamma, W)$, is computed as follows.

DOI: 10.1002/cpe

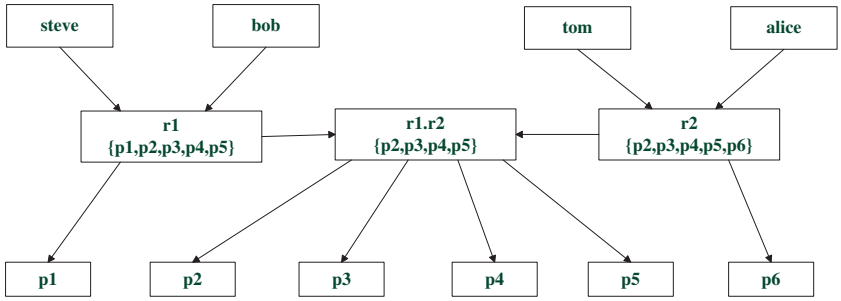


Figure 2. Two roles that have a common set of permissions.

$$wsc(\gamma, W) = wr*|\mathrm{R}| + wu*|\mathrm{UA}| + wp*|\mathrm{PA}| + wh*|\mathrm{t\_reduce(RH)}| + wd*|\mathrm{DUPA}|$$

where $|\cdot|$ denotes the size of the set or relation, and t_reduce(RH) denotes the transitive reduction of role hierarchy.

A transitive reduction is the minimal set of relationships that encodes the same hierarchy. For example, t_reduce ($\{(r\mathbf{1}, r\mathbf{2}), (r\mathbf{2}, r\mathbf{3}), (r\mathbf{1}, r\mathbf{3})\}$) = $\{(r\mathbf{1}, r\mathbf{2}), (r\mathbf{2}, r\mathbf{3}),\}(r\mathbf{1}, r\mathbf{3})$ can be inferred.

Arithmetic involving ∞ is defined as follows. $0 * \infty = 0$, $\forall x \in \mathrm{N} + x * \infty = \infty$, $\forall x \in \mathrm{N} \cup \{\infty\}\, x + \infty = \infty$. Intuitively, in role mining, we would like to find an RBAC state that has the smallest weighted structural complexity. One can adjust the weights to limit the RBAC states to be considered and to meet different optimization objectives.

In this paper, we mainly consider two classic and widely used weight schemes in the role mining field, $W1 : wr = wu = wp = wh = wd = 1$, and $W2 : wr = wu = 1$, $wp = wh = wd = 2$. The first thinks that the relation between roles is as important as the number of role or user while the latter thinks the relation between roles is more important. Because our algorithm can wholly cover the user-permission-assigned relation, in fact, in two schemes, the *wd* is equal to 0.

4.3. Role mining algorithm framework

In this section, we will present a role mining algorithm based on cardinality constraints (CCR). CCR mainly includes three steps: generating the initial role set, selecting role pair for role update algorithm, and updating the initial role state. Because the initial role state does not have hierarchical relationships and its assigned cost may be high, we update the role state according to role update algorithm for getting hierarchical and low-cost role state. When updating the role state, we may have many role pairs that can be handled in the role update process. We need to use role selection algorithm to select two roles that will be handled in the next step. After updating the role state, we need to check whether the initial constraints under the initial user-permission assignment are the solution. CCR will output the solution. Otherwise, it should tell the case if there is no solution.

The algorithm for CCR is presented in Algorithm 1. For the sake of simplicity, we use $\mathrm{P}(u_i)$ to represent the assigned_permissions (u_i) and $\mathrm{P}(r_i)$ to represent the assigned_permissions(r_i). In the beginning, we generate a role for each user's permissions (lines 4–11). If a user's cardinality constraints of roles are one, the role will be one of the final role sets (line 8). We eliminate the users whose cardinality constraints of roles are one and their permissions. Then, we will sort the roles by the number of permissions of each role in descending order (line 12), which is to make the role update algorithm converge faster. In line 13, we update the permissions' cardinality constraints according to the operation role sets *OpRSet* (shown in Algorithm 2). Lines 15–27 deal with three cases of the aforementioned. Lines 29–33 judge whether there is a solution in the current condition. If these roles that are acquired according to the cardinality constraints of roles do not satisfy the cardinality constraints of permissions, there is no solution. For the sake of simplicity, we delete the roles that have the same

DOI: 10.1002/cpe

permissions and only leave one whose constraint is the strictest. The situation of subset and intersection may also appear in the beginning or middle of the algorithm. Therefore, after processing the equation, we also immediately process the subset and the intersection.

In Algorithm 1, lines 18–22 select the best role state according to the optimal objective function *wsc*. If the current operation makes the *wsc* increase, we undo it. Lines 23–26 check whether there are some roles whose constraints are one after some step. If so, we get a final role and remove it from the operator role sets. Line 34 puts the last role sets to final role sets and line 35 returns the last role state.

Algorithm 1: CCR: role mining based on cardinality constraints

Input: users u, permissions p and user-permission assignments *UPA*
Input: weight factors for complexity, $W(w_r,w_u,w_p,w_h)$
Input: M, the number of users
Input: N, the number of permissions
Output: the final role set *Rset* and role state r_{rcl}
1: generate max_roles(u) for every user in u
2: generate max_roles(p) for every permission in p
3: {generate the initial role sets}
4: **for** $i = 0...M$ **do**
5: create a role r_i that has P(u_i) permission set
6: add(*OpRSet,* r_i)
7: **if** max_roles(u_i)= 1 **then**
8: add(*Rset,* r_i)
9: remove(*OpRSet,* r_i)
10: **end if**
11: **end for**
12: sort the roles by the number of permissions of each role in descending order
13: UpdatePermissionConstraints(*OpRSet,p*)
14: {merge roles until stable}
15: **while** role state is not stable **do**
16: identify two roles according to role selection algorithm
17: update current role state according to role update algorithm
18: **if** new *wsc* < old *wsc* **then**
19: old *wsc* = new *wsc*
20: **else**
21: undo operation
22: **end if**
23: **if** max_roles(p_i)= 1 **then**
24: add(*Rset,* r_i)
25: remove(*OpRSet,* r_i)
26: **end if**
27: **end while**
28: {judge whether there is the solution}
29: **for** $i = 0..N$ **do**
30: **if**(max_roles(p]]>$_i$) < 0)**then**
31: return no solution
32: **end if**
33: **end for**
34: append *OpRSet* to the end of *Rset*
35: return *Rset* and r_{rcl}

DOI: 10.1002/cpe

Algorithm 2: UpdatePermsConstraints(*Tset,p*)

Input: *TSet,* the role sets
Input: permissions *p*
Input: *N*, the number of permissions
1: **for** *i = 0…size(TSet)* **do**
2: **for** *j = 0…N* **do**
3: **if** $p_j \subseteq P(r_i$ **then**
4: max_roles (p_j) = max_roles (p_j) – 1
5: **end if**
6: **end for**
7: **end for**
8: return max(P)

4.4. Role selection algorithm

Role selection algorithm is used in line 16 of Algorithm 1. In role selection algorithm, we first sort the roles by the number of permissions of each role in descending order. If two roles have the same number of permissions, we sort the two roles by the number of users of the role in descending order. Even if two roles have the same number of permissions and the same number of users in the role, we select the role with hierarchy. If both the two roles do not have hierarchies, we select one role randomly. After sorting roles, we select two roles at the beginning of the role set.

The reason of sorting the role set is to make the role update algorithm faster convergence. When we put these roles with more permissions in the front, the new roles with more permissions will generate priority in the process of subset operation and intersected operation in role update algorithm. While handling subset and intersection, we need to add the new roles into the original role set. If the original role set is sorted, we only need to insert the current role into the right position. We do not need to sort the role set again. Hence, we transform the time complexity from $O(n*\log n)$ to $O(n)$, and n is the number of original role set.

4.5. Role update algorithm

In [11], Zhang et al. divide the graph optimization algorithm into four scenarios. In our algorithm, we also consider four scenarios, but each scenario has different operations. Our algorithm does the pretreatment in scenario 1. In this scenario, there are two users under current investigation who have the same permissions. Then, the two users' roles will be merged and the cardinality constraint of the new role is the smaller one in the cardinality constraints of two original roles (Figure 3) if the cardinality constraints of two original roles are not one.

Figure 3 shows the scenario that two users' permission sets are the same. Figure 3(a) shows the situation before merging. In the figure, the first column of the frame is the index of the role in current role sets; the second column is the user sets of the current role. The upper of the third column is the permission sets of the current role and the bottom is cardinality constraints of the corresponding permission. If the bottom is empty, it illustrates that there is no constraints for the corresponding permission. The last column is the current users' cardinality constraints of roles. Figure 3(b) shows the result of the merging. From the above description, we know the role whose constraint is less will be deleted. At the same time, the index is the smaller one of the two parents and the users are the sum of two parents. The number of cardinality constraints on each permission is the larger one in the cardinality constraints of two original roles. At last, the current users' number of cardinality constraints of roles is the smaller one of the two parents. We can also see that the *wsc* will reduce in this process because we delete a role and its relations.

Scenario 2 is that the permission sets of the first role are a subset or superset of the permission sets of the second role (Figure 4). In this case, an edge is created from the superset permission role to the subset permission role. In order to overcome the drawback in [11], we delete the common permissions from the superset permission role. By doing so, we can assign the subset permission

DOI: 10.1002/cpe

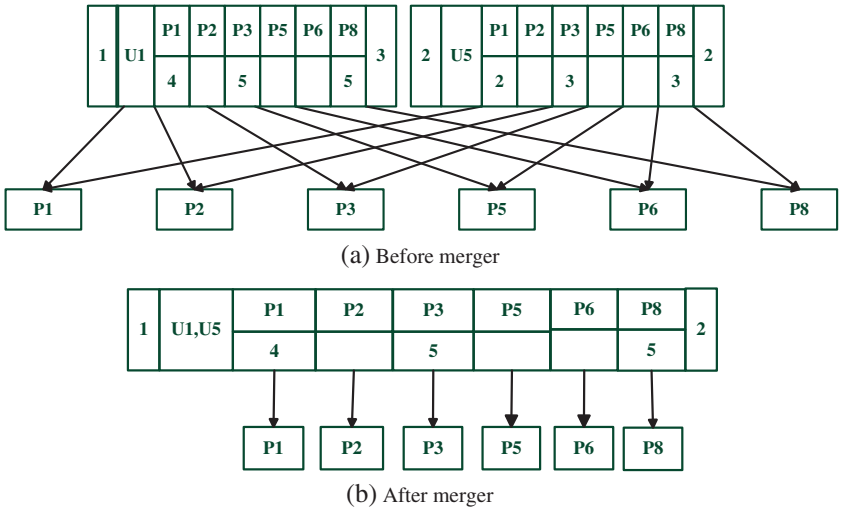


Figure 3. Merging of two roles that have the same permissions. (a) Before merger and (b) after merger.

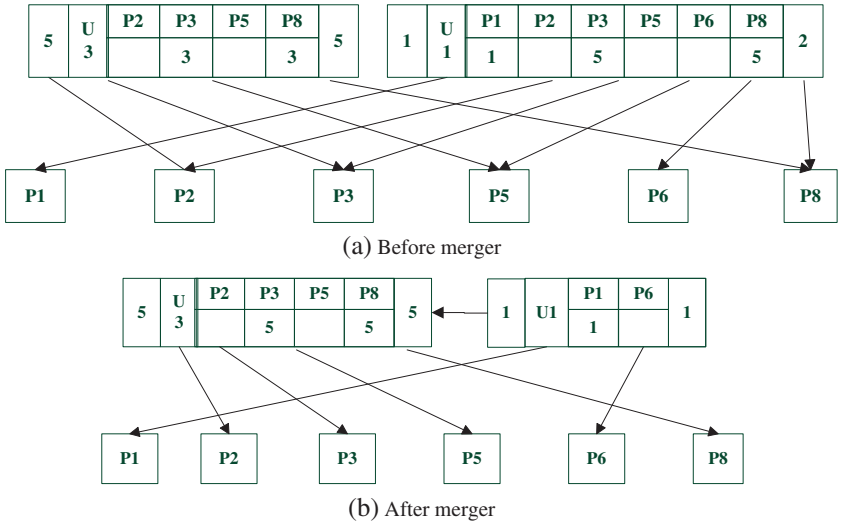


Figure 4. One role's permissions are the superset of another role's permissions. (a) Before merger and (b) after merger.

role to user $u1$. Of course, the previous assigned method has been deleted. For instance, in Figure 4(a), the way that the number one role is assigned to user $u1$ will be deleted. If the number of subset permissions is greater than one, the *wsc* will decrease or will have no effect. Because the number of superset permissions decreases by one and the number of roles increases by one, the cardinality constraints on each permission are stable. At last, the cardinality constraints of parent roles decrease by one.

The third situation is illustrated in Figure 5, where two roles have a common set of permissions. Figure 5(a) shows the scene before merging and the operation result is presented in Figure 5(b). Under the situation, a new role is created that owns the common permissions. At the same time, two edges are also created from the two original roles to the new role. The max_roles(p) in the new role is the larger number between the two parent roles. As the second situation, we delete the common permissions from two parent roles. At last, because we split two parent permission set, the max_roles(u) of two parent roles decreases by one. The max_roles(u) of the new role is the smaller between two parent roles that have been handled. Identically, if the number of two current roles' common permissions is greater than three, the *wsc* becomes smaller.

DOI: 10.1002/cpe

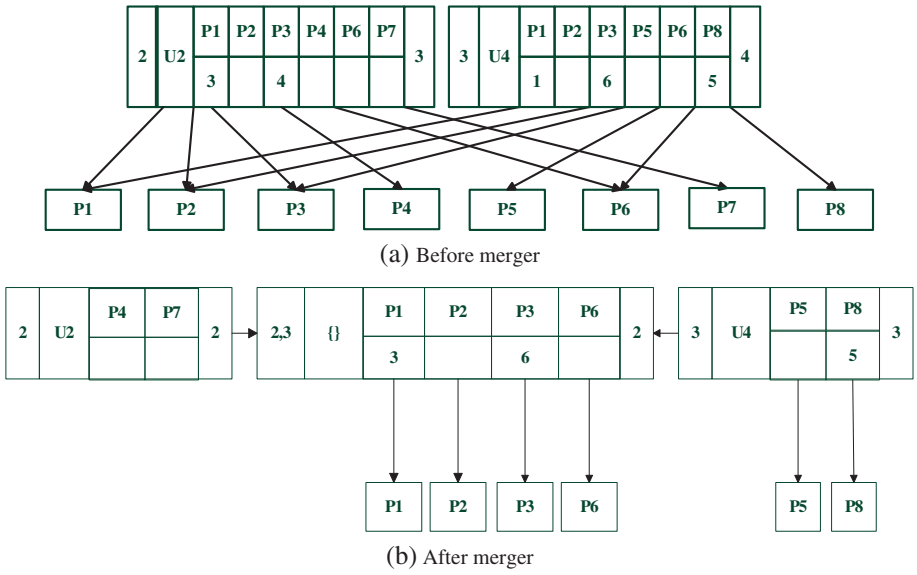


Figure 5. Two roles that have a common set of permissions. (a) Before merger and (b) after merger.

In the second and third cases, when the number of common permissions or the permissions of the subset role is greater than three, the *wsc* will become lower, because we increase a role and five hierarchy edges. At the same time, we reduce six edges from the parent roles to the permissions directly. Therefore, we transform the problem of reducing the *wsc* to judge whether the number of the common permissions is greater than three.

The last case is that two roles do not have any relation. In this case, if we merge and cross the role's permissions in the aforementioned way, the *wsc* will increase. Hence, we do nothing.

4.6. Computational complexity

Because the CCR is based on the GO algorithm, the increased operations do not influence the time complexity. Hence, the time complexity of the CCR is the same as the GO. It is $O(n^2m)$, where n is the number of users and m is the number of permissions [27]. However, the CCR also has the shortcoming that the GO has. It is unclear how many iterations the algorithm would need before stopping. Suppose we set the number of iterations to k, the time complexity is $O((k+n)km)$. Therefore, we can set the number of iterations to control the running time.

5. EXPERIMENTAL RESULTS

5.1. Generate constraint data

To ensure only the users who have a certain number of permissions own the cardinality constraints in most situations, we define the rules as follows.

Definition 6
The average number of permissions owned by a user is defined as

$$wnp(\mathrm{U}) = \frac{\sum_{i=1}^{\mathrm{numUsers(All)}} \left|\mathrm{assigned}_{\mathrm{permissions}}(ui)\right|}{\mathrm{numUsers(All)}}$$

where numUser(All) is the number of all users. Similarly, we can define the weighted number of users.

DOI: 10.1002/cpe

Definition 7
The average number of users owned by a permission is defined as

$$wnu(\mathrm{P}) = \frac{\sum_{i=1}^{\mathrm{numPermissions(All)}} |\mathrm{assigned}_{\mathrm{users}}(pi)|}{\mathrm{numPermissions(All)}}$$

where numPermissions(All) is the number of all permissions.

According to Definition 6, we can define the maximal number of permissions owned by a user. We can employ the user cardinality constraints of roles to restrict the number of permissions owned by a user.

Definition 8
The max number of permissions a user can own is defined as (if more, it will give a constraint)

$$\mathrm{permUpperBound} = \left\lceil \frac{\max(\mathrm{PermissionCount(All)})}{wnp(U)} \right\rceil * \alpha * wnp(U)$$

where α is a weighted factor between 0 and 1, denoting the extent of the limits; max(PermissionCount (All)) is the number of permissions the user owns in the UPA. From Definition 8, we can see that the smaller the α is, the smaller the permUpperBound is. While permUpperBound is small, the number of cardinality constraints of users will increase.

Similarly, we can define the maximal number of users that the permissions can belong to as follows.

Definition 9
The maximal number of users that the permission can belong to is defined as

$$\mathrm{userUpperBound} = \left\lceil \frac{\max(\mathrm{UsersCount(All)})}{wnu(P)} \right\rceil * \beta * wnu(P)$$

where the meaning of β is similar with α. We can employ the cardinality constraints of users to restrict the number of users that the permission can be assigned to.

If a user has the number of permissions larger than permUpperBound, we will generate a number that is between 1 and permUpperBound. That is to say, we will map the range of assigned_permissions(u) to permUpperBound.

To calculate the cardinality constraints in the experiment, we need to generate a positive integer k that is between 1 and permUpperBound. If k is closer to permUpperBound, the probability is bigger. On the contrary, if k is closer to 1, the probability is smaller. The relationship shown in Figure 6 satisfies all above requirements. We use the relationship in the experiments. Meanwhile, according to probability theory, we must also make sure that the sum of probability generating from 1 to k is 1. For the sake of simplicity, in the rest of this paper, we use x to represent the permUpperBound and z to represent the assigned_permissions(u).

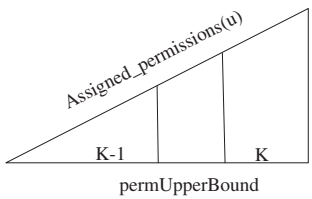


Figure 6. Map assigned_permissions(u) to permUpperBound.

DOI: 10.1002/cpe

Lemma 1
A probability function that satisfies the aforementioned constraints is

$$\mathrm{p}(k) = \frac{2k-1}{x^2} \ (k \in (1, \ \mathrm{permUpperBound}))$$

Proof
From Figure 2, we can easily get the hypotenuse equation of the triangle as follows

$$\mathrm{f}(x) = \frac{\sqrt{z^2 - x^2}}{x}$$

If we express the probability with the area, when x is equal to k, the probability is

$$\mathrm{p}(k) = \frac{\frac{\sqrt{z^2-x^2}}{x} * k^2 * \frac{1}{2} - \sum_{i=1}^{k-1} pi}{\sqrt{z^2 - x^2} * x * \frac{1}{2}} \tag{1}$$

In fact, $\mathrm{p}(k)$ is the area of the trapezoid between $k-1$ and k. According to the front hypothesis that the area represents the probability, we can get

$$\sum_{i=1}^{k-1} pi = \frac{\sqrt{z^2 - x^2} * (k-1)^2 * \frac{1}{2}}{x}.$$

Hence, Equation (1) is

$$\frac{2k-1}{x^2}.$$

The lemma illustrates that the probability generating k is $\frac{2k-1}{x^2}$.

Lemma 2
The sum of probability in Lemma 1 is 1.

Proof
The sum of probability in Lemma 1 is

$$\frac{1 + 3 + 5 + \ldots + (2x-1)}{x^2} = \frac{2x * x * \frac{1}{2}}{x^2} = 1.$$

Similarly, we can get the probability function of generating permissions as follows

$$\mathrm{p}(k) = \frac{2k-1}{x^2} \ (k \in (1, \ \mathrm{userUpperBound}))$$

where x represents userUpperBound.

Of course, the aforementioned is effective if the system users do not appoint the constraints, or we will use these constraints that are appointed by the system users. That is to say, if we get the role

DOI: 10.1002/cpe

constraint of the U_i, denoted as c_1, through the above process, and the constraint is c_0 appointed by the system users, we can get the weighted cardinality constraints as follows.

Definition 10
The weighted cardinality constraint of a user is

$$c = r * c_1 + d * c_0$$

where r and d are either 0 or 1, and r XOR $d = 1$. If there is no presettings, we set $r = 1$ and $d = 0$.

5.2. Experimental preparation

To evaluate the proposed approach, we implement our algorithm on a Dell OptiPlex 520 PC with Intel Pentium (R) D 2.80GHz CPU and 1GB memory made in China. We run our algorithm on real data sets in Table I.[2] In the table, |U| represents total number of users. |P| represents total number of permissions available. |UPA| represents total number of permissions assigned to all users, that is, the size of the UPA matrix. Density represents what percentage of total assignable permissions is actually assigned to the users of the system in the UPA.

We compare the assignment cost with other current role mining algorithms as the constraints change. That is to say, α and β change from 0 to 1. Meanwhile, we compare accuracy with *HierarchicalMiner* under no constraints in our algorithm. We define the accuracy as follows.

Definition 11
The accuracy of the algorithm is defined as

$$\mathrm{ACC} = \frac{\mathrm{numRoles(Match)}}{\mathrm{numRoles(Match)} + \mathrm{numRoles(notMatch)}}$$

where numRoles(Match) is the number of exact match between the original role sets and the generated role set, and numRoles(notMatch) is the number of roles that do not match. The number of notMatch contains two parts: the original role set that is not matched and the generated role set that is not matched. In the practical application, we can set weight to illustrate the two parts' importance of degree. For the sake of simplicity, we simply set them the same weight in this paper. Hence, we can add them together and express it with the above definition.

In [20], Ma et al. define the accuracy with the ratio of the number of match to the number of original role set. If using the aforementioned definition, it can enumerate all possible roles with the *CompleteMiner*. However, it not only generates too many roles, but the process takes too much time. Therefore, we will not use their formula in our paper.

Table I. Experimental data sets.

Dataset	|U|	|P|	|UPA|	Density (%)
Healthcare	46	46	1486	70
Domino	79	231	730	4
EMEA	35	3046	7220	6.8
Firewall1	365	709	31,951	12.3
Firewall2	325	590	36,428	19
University1	493	56	3955	14.3
University2	400	14	3073	54.9

|U|, total number of users; |P|, total number of permissions available; |UPA|, total number of permissions assigned to all users.

DOI: 10.1002/cpe

5.3. Experimental evaluation

5.3.1. Accuracy. To test the accuracy, we use the university2 dataset [12], which is also used in [9] and has 400 users and 14 permissions. Because this dataset contains an original role set, we use the dataset to test the accuracy. To check out the WSC, we use the other datasets in Table I. Table II shows how the number of constraints and the accuracy change as the role limits upperBound (α) change when β is equal to 0.9. It also shows how the number of constraints and the accuracy change as the permission limits upperBound (β) change when α is equal to 0.8. In the table, RC represents the number of generated role constraints, PC represents the number of generated permission constraints, and ACC represents the accuracy. We use the same symbols in the rest of this paper.

The result of Table II is illustrated further in Figure 7. From Table II and Figure 7, we can see that the role constraints have greater influence on the accuracy than the permission constraints. That is mainly because there are more users than permissions in the university2 dataset. Figure 8 shows the result of no constraints, that is to say, both α and β are equal to 1.0. Figure 8(a) is the original role set, Figure 8(b) is the result of GO algorithm, and Figure 8(c) is the result of our proposed CCR algorithm when there are no constraints. We can easily see that all roles except *Grader* exactly match the corresponding original role set in the CCR. Nevertheless, in fact, *Grader* also matches the original role *Grader* because it automatically gets the permission of *viewGrade_GradeBook* from the role *Student*. Therefore, all the roles obtained by CCR are exactly matched. Besides, the permissions of the role *Grader* are the same as *TA* in the original role set. In fact, there are only nine roles in the original role set.

Table II. Number of constraints and accuracy change as the role limits change.

(a) Number of constraints and accuracy as α changes

	RC	PC	ACC
$\alpha = 0.2$	400	5	0.3
$\alpha = 0.4$	400	5	0.6
$\alpha = 0.6$	85	5	0.7
$\alpha = 0.8$	9	5	0.8
$\alpha = 1.0$	0	5	0.8

(b) Number of constraints and accuracy as β changes

	RC	PC	ACC
$\beta = 0.2$	9	10	0.8
$\beta = 0.4$	9	7	0.8
$\beta = 0.6$	9	7	0.8
$\beta = 0.8$	9	5	0.8
$\beta = 1.0$	9	0	0.8

RC, number of generated role constraints; PC, number of generated permission constraints; ACC, accuracy.

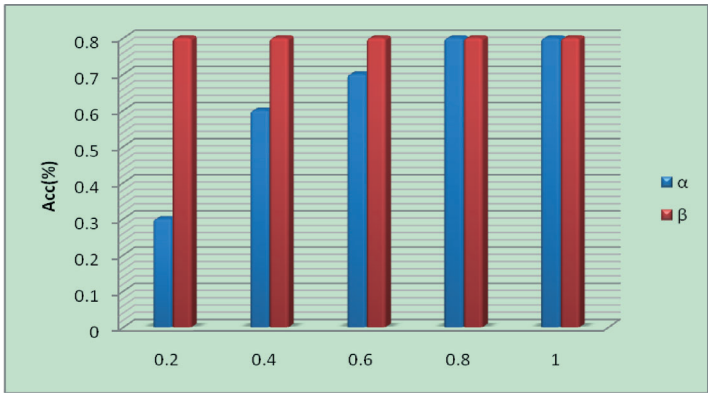
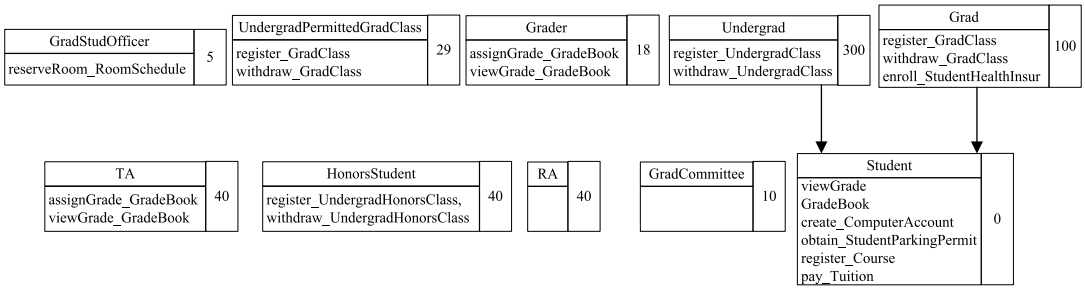
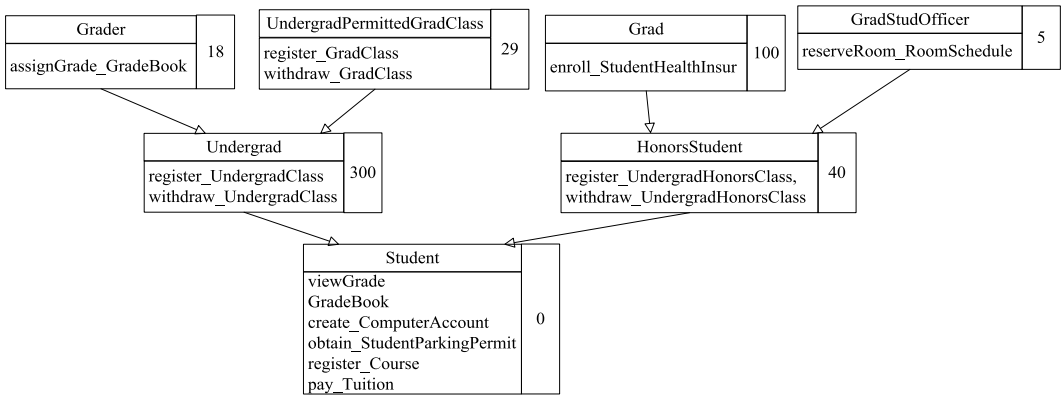


Figure 7. Accuracy change as the role constraint (α) and permission constraint (β).

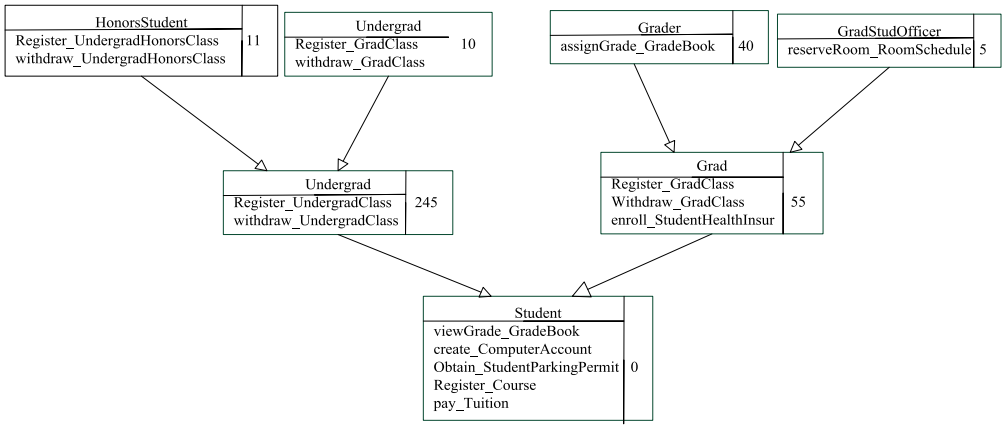
DOI: 10.1002/cpe



(a) Original role set



(b) Role set generated by GO



(c) Role set generated by CCR

Figure 8. The result of cardinality constraints (CCR) under no constraints. (a) Role set generated by graphic optimization (GO) and (b) role set generated by CCR.

According to Definition 11, we can get the accuracy of CCR under the condition of no constraints, which is 7/9 = 77.8%. In the equation, the number of roles exactly matched is seven, the number of roles that are not matched in the original role set is two, and the number of roles that are not matched in the generated role set is zero. When the number of cardinality constraints of roles changes, the roles exactly matched may become not matched. Nevertheless, at least the roles *Undergrad* and *Student* are exactly matched.

Similarly, according to Definition 11, we can get the accuracy of GO, which is 6/10 = 60%. In the equation, the number of roles exactly matched is six except *Grad*. The number of roles that are not matched in the original role set is two, and the number of roles that are not matched in the generated role set is one (i.e. *Grad*). Therefore, our algorithm CCR increases the accuracy by 29.7% (from 60% to 77.8%).

DOI: 10.1002/cpe

5.3.2. Assignment cost. On the dataset university1, Table III shows how the number of constraints and WSC change as the role constraints upperBound(α) changes when β is equal to 0.9. It also shows how the number of constraints and WSC change as permission constraints upperBound(β) changes when α is equal to 0.8. Figure 9 further shows the trend of changes. We can see that the WSC is about the same with *GraphOptimization* in [12] when the number of constraints is not too large.

From Table III and Figure 9, we know that the number of role constraints is larger than the number of permission constraints when α or β changes in the same range. This is mainly because there are more users than permissions in the test dataset. That is also the reason why α is equal to 0.8 when β changes and β is equal to 0.9 when α changes. From Table III and Figure 9, we can also see that the WSC has little close relation to the role constraints, and the value of WSC changes little. We can also see that the WSC is less variable on the permission constraints than the role constraints. That is also mainly because there are more users than permissions in the dataset. We also see that the WSC is the same magnitude with the *GraphOptimization* in [11] whose WSC is equal to 615.

In [14], John et al. only consider the cardinality constraints of roles and propose two algorithms, RPA and CPA. Hence, we set the parameter of the cardinality constraints of permissions β to 1.0 in our experiment. That is, we only consider the factor of the cardinality constraints of roles α. We run CCR on the datasets in Table I, and plot the variation of the number of roles (y-axis) generated by

Table III. Number of constraints and WSC change as the limits change.

(a) Number of constraints and WSC as α changes

	RC	PC	R	UA	PA	RH	WSC
$\alpha=0.2$	158	1	20	493	103	24	640
$\alpha=0.4$	10	1	18	493	64	30	605
$\alpha=0.6$	6	1	18	493	64	31	606
$\alpha=0.8$	1	1	18	493	64	36	611
$\alpha=1.0$	0	1	18	493	64	38	613

(b) Number of constraints and WSC as β changes

	RC	PC	R	UA	PA	RH	WSC
$\beta=0.2$	1	11	18	493	64	36	611
$\beta=0.4$	1	7	18	493	64	36	611
$\beta=0.6$	1	7	18	493	64	36	611
$\beta=0.8$	1	5	18	493	64	36	611
$\beta=1.0$	1	0	18	493	64	36	611

WSC, weighted structural complexity; RC, number of generated role constraints; PC, number of generated permission constraints; UA, user-role assignment; PA, permission-role assignment; RH, role hierarchy.

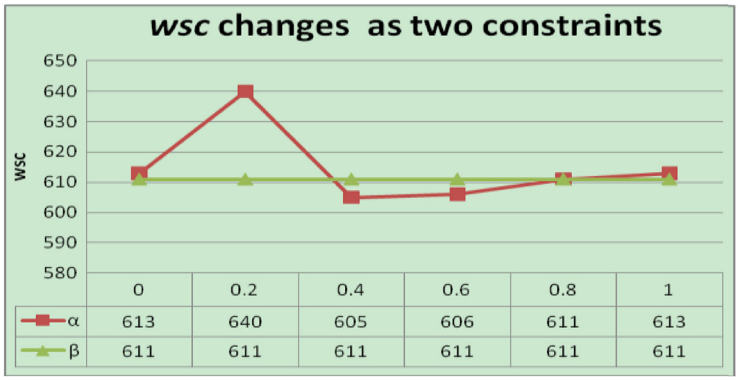
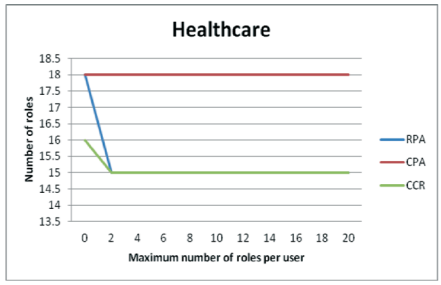
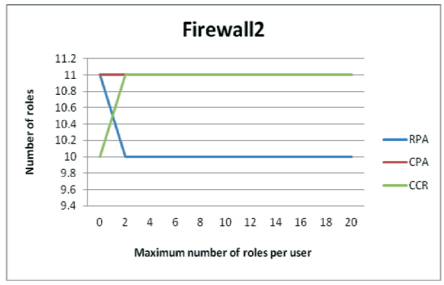


Figure 9. Weighted structural complexity (WSC) changes as the role constraint (α) and permission constraint (β).

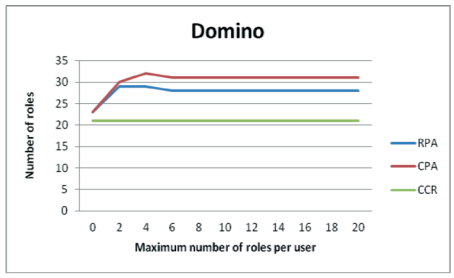
DOI: 10.1002/cpe



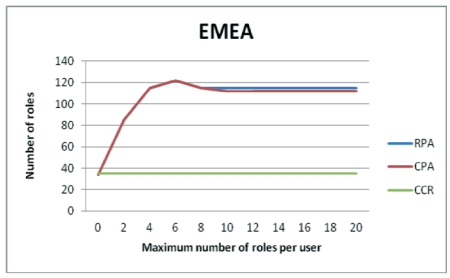
(a) Number of roles for Healthcare



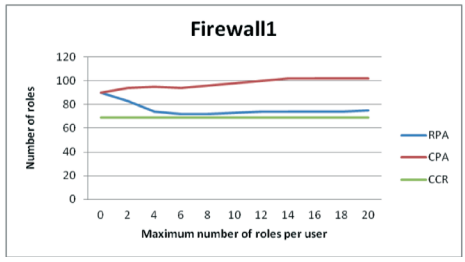
(b) Number of roles for Firewall2



(c) Number of roles for Domino



(d) Number of roles for EMEA



(e) Number of roles for Firewall1

Figure 10. The number of roles discovered by three different algorithms for (a) Healthcare, (b) Firewall2, (c) Domino, (d) EMEA, and (e) Firewall1. RPA, role-priority-based algorithm; CPA, coverage of permissions-based algorithm; CCR, cardinality constraints.

RPA, CPA, and CCR while varying the constraint value (x-axis). The roles discovered by the three algorithms satisfy the given constraints. The plots are shown in Figure 10(a)–(e).

From these figures, we can see that the number of roles generated by CCR is the least except the firewall2 dataset among these three algorithms. The result of CCR is only one more than CPA on the firewall2. Although the RPA decomposes the UPA matrix, it is not the full rank decomposition. Therefore, the RPA generates lots of roles. Because the CPA generates the initial role set with the way of FastMiner, and FastMiner enumerates all possible roles, it also generates many roles. Moreover, the results of CPA and RPA have no role hierarchy. Our algorithm is based on graph optimization theory, and the initial role set is generated by each user's permission set. Then, we update the initial role set based on graph optimization theory. Therefore, the generated roles of CCR are hierarchical. Because the number of users is limited, our algorithm CCR generates the least role set.

Through the comparison between our algorithm and other role mining algorithms in terms of the accuracy and WSC, we can see that our algorithm works better under the constraints.

6. CONCLUSIONS AND FUTURE WORK

It is important to take the constraints into account in the RBAC and the process of role mining. In this paper, we focus on cardinality constraints and analyze the problem of the cardinality constraints of

DOI: 10.1002/cpe

roles and permissions. We then formally define the problem and generate the constraints in the real world dataset. We present an algorithm to deal with the role mining problem under the two constraints. The algorithm is based on the graph optimization theory. In the paper, we point out the problem of the existing approaches in which some roles are found but cannot be assigned to any user. Our algorithm can fix this problem. We also verify the effectiveness of the proposed algorithm through experiments. Our future work is to design an algorithm that can solve all cardinality constraint problems. Besides this, there are different contexts for the constraints in the role mining process, such as the prerequisite role constraints. We plan to consider these different contexts of constraints in the role mining study.

ACKNOWLEDGEMENTS

This work is supported by the National Natural Science Foundation of China under Grants 61173170, 61300222, 61433006 and U1401258, the National High Technology Research and Development Program of China under Grant 2007AA01Z403, and the Innovation Fund of Huazhong University of Science and Technology under Grants 2013QN120, 2012TS052 and 2012TS053.

REFERENCES

1. Ferraiolo D, Sandhu R, Gavrila S, Kuhn D, Chandramouli R. Proposed NIST standard for role-based access control. *ACM Transactions on Information and System Security* 2001; **4**(3):224–274.
2. Sandhu RS, Coyne EJ, Feinstein HL, Youman CE. Role-based access control models. *IEEE Computer* 1996; **29**(2):38–47.
3. Baumgrass A, Strembeck M, Ma SR. Deriving role engineering artifacts from business processes and scenario models. In Proc. ACM Symposium on Access Control Models and Technologies (SACMAT). New York USA: ACM, 2011. 11–20.
4. Strembeck M. A role engineering tool for role-based access control. In Proc. of the 3rd Symposium on Requirements Engineering for Information Security (SREIS). Paris France,August 2005. 7–14.
5. Strembeck M. Scenario-driven role engineering. *IEEE Security&Privacy* 2010; **8**(1):28–35.
6. Strembeck M, Neumann G. An integrated approach to engineer and enforce context constraints in RBAC environments. *ACM Transactions on Information and System Security (TISSEC)* 2004; **7**(3):392–427.
7. Fernandez EB, Hawkins JC. Determining role rights from use cases. In Proceedings of the 2th ACM Workshop on Role-Based Access Control. New York USA:ACM. 1997. 121–125.
8. Vaidya J, Atluri V, Warner J. RoleMiner: mining roles using subset enumeration. In Proc. ACM Conference on Computer and Communications Security (CCS).New York USA: ACM, 2006. 144–153.
9. Zhang D, Ramamohanarao K, Ebringer T, Yann T. Permission set mining: discovering practical and useful roles. In Proceedings of the 2008 Annual Computer Security Applications Conference (ACSAC). IEEE Computer Society: Washington USA, 2008; 247–256.
10. Schlegelmilch J, Steffens U. Role mining with ORCA. In Proc. ACM Symposium on Access Control Models and Technologies (SACMAT). ACM: New York USA, 2005; 168–176.
11. Zhang D, Ramamohanarao K, Ebringer T. Role engineering using graph optimisation. In Proc. ACM Symposium on Access Control Models and Technologies (SACMAT. ACM: New York USA, 2007; 139–144.
12. Molloy I, Chen H, Li T, Wang Q, Li N, Bertino E, Calo S, Lobo J. Mining Roles With Semantic Meanings. In Proc. ACM Symposium on Access Control Models and Technologies (SACMAT), Pages 21–30, 2008.
13. Frank M, Buhmann JM, Basin D. On the definition of role mining. In Proc. ACM Symposium on Access Control Models and Technologies (SACMAT). ACM:: New York USA, 2010; 35–44.
14. John C, John SS, Vijayalakshmi A, *et al.* Role mining under role-usage cardinality constraint. In 27th IFIP International Information Security and Privacy Conference. Springer Berlin Heidelberg: Crete Greece, 2012; 150–161.
15. G Neumann and M Strembeck. A scenario-driven role engineering process for functional RBAC roles. In Proc. ACM Symposium on Access Control Models and Technologies(SACMAT). New York USA:ACM, 2002.33–42.
16. Zhang D, Ramamohanarao K, Versteeg S. Graph based strategies to role engineering. In Proceedings of the Sixth Annual Workshop on Cyber Security and Information Intelligence Research.New York USA:ACM, 2010. 1–4.
17. Vaidya J, Atluri V, Guo Q. Migrating to optimal RBAC with minimal perturbation. In Proc. ACM Symposium on Access Control Models and Technologies (SACMAT). ACM: New York USA, 2008; 11–20.
18. J Vaidya, V Atluri, and Q Guo. The role mining problem: finding a minimal descriptive set of roles. In Proc. of the 12th ACM symposium on Access control models and technologies (SACMAT).New York USA: ACM,2007.175–184.
19. H Lu, J Vaidya, and V Atluri. Optimal boolean matrix decomposition: application to role engineering. In Proc. of the IEEE 24th International Conference on Data Engineering (ICDE). Washington USA: IEEE Computer Society, 2008.297–306.
20. Ma X, Li R, Lu Z. Role mining based on weights. In Proc. of ACM Symposium on Access Control Models and Technologies (SACMAT). ACM: New York USA, 2010; 65–74.

DOI: 10.1002/cpe

21. Takabi H, Joshi JBD. StateMiner: an efficient similarity-based approach for optimal mining of role hierarchy. In Proc. of ACM Symposium on Access Control Models and Technologies (SACMAT). ACM: Pittsburgh USA, 2010; 55–64.
22. Molloy I, Li N, Lobo J, Dickens L. Mining roles with noisy data. In Proc. ACM Symposium on Access Control Models and Technologies (SACMAT). ACM: New York USA, 2010; 45–54.
23. Lu H, Vaidya J, Atluri V, Hong Y. Constraint-aware role mining via extended boolean matrix decomposition. IEEE Transactions on Dependable and Secure Computing 2012; **9**(5):655–669.
24. Kumar R, Sural S, Gupta A. Mining RBAC roles under cardinality constraint. In Proceedings of the 6th International Conference on Information Systems Security (ICISS'10). Springer-Verlag: Gandhinagar India, 2011; 171–185.
25. Ferraiolo D, Sandhu R, Gavrila S, Kuhn D, Chandramouli R. Proposed NIST standard for role-based access control. *ACM Transactions on Information and System Security (TISSEC)* 2001; **4**(3):224–274.
26. Ma X, Li R, Lu Z. Mining constraints in role-based access control. *Mathematical and Computer Modelling* 2012; **55**(1-2):87–96.
27. Molloy I, Li N, Li T, Mao Z, Wang Q, Lobo J. Evaluating role mining algorithms. In Proceedings of the 14th ACM Symposium on Access Control Models and Technologies (SACMAT). ACM: New York USA, 2009; 95–104.

DOI: 10.1002/cpe